



# PLANT GROWTH SYSTEM

## BASIC USER MANUAL

<b>Version:</b> 1.0	<b>Date:</b> October 9, 2025
<b>Project Title:</b> Plant Growth System	<b>Document:</b> Basic User Manual

### CONTENT

- Scripts.
- Demo Scene
- Example Prefab.

### GAMEOBJECT SCRIPTS

- Soil
- PlantEntity
- PlantState
- Weather
- PlantGrowthSystemConfig

## INTRODUCTION

### Purpose of the System

This plugin is a **plant growth simulation system** for Unity, designed to give developers full control over plant behavior and interactions in their scenes. It allows you to create configurable plant prefabs that can be used in farming simulations, survival games, or any environment where dynamic plant systems are needed.

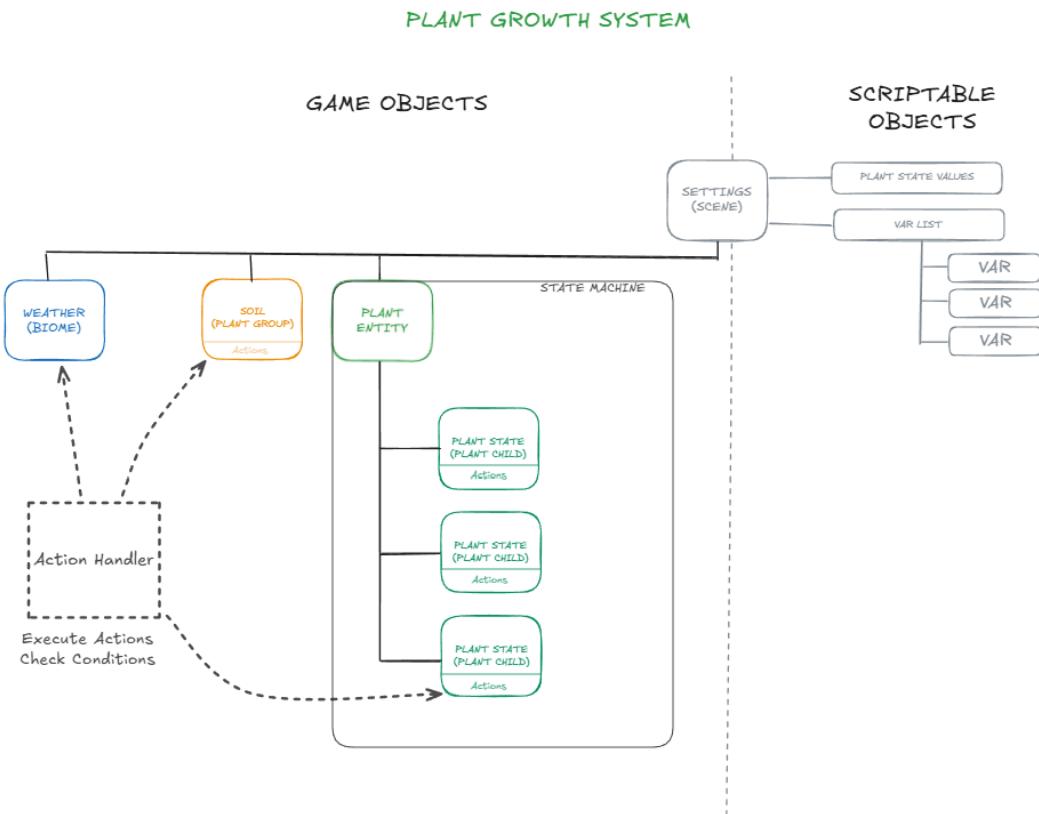
The system is built around three core scripts: **Weather**, **Soil**, and **PlantEntity**. Each **PlantEntity** contains one or more **PlantStates**, representing different aspects of the plant's growth and condition. All of these scripts are interconnected through **VARS** – configurable physical magnitudes such as **Moisture**, **Biomass**, **Sunlight**, **Temperature**, and more.

**VARS** define the “recipe” for each configuration, while **VariableStates** represent the current state of a VAR for a specific plant. These VariableStates drive **actions** that can be executed either automatically via a configurable time-based cycle (tick), by conditional triggers, or manually.

This structure allows you to **fully customize plant behavior** by defining variables, states, conditions, and actions within a prefab. Once configured, plants can be instantiated anywhere in your scene, maintaining **persistent state data** at runtime and enabling complex interactions with other systems.

In essence, this plugin provides a **state-machine-based system** with prefab-level configurability and external data persistence, empowering developers to create rich, dynamic plant simulations without writing custom scripts from scratch.

## Key Concept



## CORE SCRIPTS OVERVIEW

### WEATHER SCRIPT

#### Explanation

The **Weather** component defines the global environmental conditions of the scene. It acts as a shared reference for all **PlantEntities**, providing biome-related data such as sunlight, temperature, or other atmospheric variables.

### SOIL SCRIPT

#### Explanation

The **Soil** component represents the ground conditions that affect plant growth. It provides local environmental variables such as **Moisture**, **Drainage**, and **Nutrients**, which directly influence the state of connected **PlantEntities**.

## PLANT\_ENTITY SCRIPT

### Explanation

The **PlantEntity** represents a single plant in the scene.

It serves as the **container for configuration** and references while delegating the execution of actions and conditions to its child **PlantStates**.

Each PlantEntity can hold multiple PlantStates, allowing a modular and flexible approach to plant simulation.

### Core Responsibilities

#### 1. Global VARS & References

- Stores the **configuration shared by all child PlantStates**, such as:
  - Connection to **Weather** and **Soil**
  - Default VARS values
  - Global settings for the plant (growth parameters, species type, etc.)
- Ensures consistency between all PlantStates of the same PlantEntity

#### 2. Child PlantStates

- Each PlantState manages a specific aspect of the plant:
  - Growth
  - Health
  - Stress
  - Any other custom state
- PlantStates **execute actions and evaluate conditions** independently, but use the **global VARS provided by the PlantEntity**

#### 3. Hierarchy & Organization

- PlantEntity acts as the **root GameObject** for a plant prefab
- Allows multiple PlantStates to coexist and interact with the same environmental data
- Maintains a clean structure in the scene for debugging and prefab management

## PLANT\_STATE SCRIPT

### Explanation

A **PlantState** represents a **biological state of the plant**, defining how the plant behaves, grows, and reacts to the environment at a given moment in its life cycle.

Each PlantState encapsulates its own logic, rules, and interactions to simulate realistic plant development.

### Biological Meaning

PlantStates are not generic logic modules.

They represent **real biological phases or conditions**, such as:

- Seed
- Sprout
- Growing
- Mature
- Stressed
- Withered
- Any custom biological state defined by the developer

Each state reflects **different biological priorities and limitations**.

### State-Specific Behavior

Within each PlantState:

- **Actions, Conditions, and VariableStates** are defined specifically for that biological state
- Growth rules, resource consumption, and reactions to Weather or Soil can differ completely from other states
- The same VAR (e.g. Moisture or Temperature) may have **different effects depending on the active PlantState**

This allows realistic transitions and behaviors.

## State Transitions

PlantStates can trigger **transitions to other PlantStates** based on Conditions.  
This enables natural life-cycle progression and reactive behavior.

Examples:

- Seed → Sprout (enough moisture and temperature)
- Growing → Stressed (lack of water)
- Mature → Withered (long-term stress)

# TUTORIAL.

## Understanding the simulator

### What do we need?

- Demo Scene

### Demo Scene Overview

#### Purpose of the Demo Scene

The Demo Scene is designed as a **controlled, minimal environment** to showcase how the core systems of the plugin interact with each other, without introducing unnecessary complexity.

#### Scene Composition

##### 1. Demo & Interface Objects

These GameObjects contain:

- Demo-specific scripts
- User interface logic
- Input handling

## 2. Global Simulation Objects

These objects represent the **core environmental systems** used by all plants in the scene.

They are instantiated **once** and act as shared references.

### **PGSGlobalSettings**

Acts as the **central configuration hub** for the plugin.

It defines:

- Global behavior rules
- System-wide constants
- Shared references used internally by PGSEntities

This object ensures that all entities operate under the same simulation assumptions.

---

### **Weather (Static)**

Defines the **global atmospheric conditions** of the scene.

It provides shared environmental data such as:

- Sunlight exposure
- Temperature
- Other biome-related variables

All PlantEntities read from the same Weather instance, ensuring **consistent environmental simulation** across the scene.

---

#### **Soil (Static)**

Represents the **ground conditions** where plants grow.

It contains data such as:

- Moisture levels
- Drainage capacity
- Soil-related modifiers affecting plant behavior

*Considerations.*

#### **Prefab-Based Assets and Datasets**

Both **Soil** and **Plant** systems are designed to be **Prefab-based**.

This means that:

- You can create multiple **Soil Prefabs**, each one representing different ground conditions (moisture, drainage, modifiers, etc.).
- You can create multiple **Plant Prefabs**, each one encapsulating its own configuration, states, and behavior.

These Prefabs are **fully reusable** and **independent from the Demo Scene**.

## **Step 1 – Add a Plant**

1. Enter Play Mode
2. Click on an existing Soil
3. Select Add Plant → Tomato Plant

The plant will be instantiated as a child of the selected Soil.

At this moment:

- ActionsOnInit are executed
- The Plant starts in its initial PlantState

## **Step 2 – Modify Variables**

- Click again on the Soil
- Select List Variables
- Locate the Moisture variable
- Increase its value

This directly affects the VariableStates shared between:

Soil, PlantEntity Current PlantState

### **Step 3 – Observe Growth Conditions**

After increasing Moisture:

- Growth-related conditions become valid
- Growth actions are executed during the simulation tick
- Plant biomass starts increasing over time

All changes happen automatically through the tick system and action evaluation loop.

### **Step 4 – State Transition**

Once the Plant reaches the required Biomass threshold:

- The current PlantState changes
- A new PlantState becomes active
- A different set of actions and conditions starts to apply

This allows each biological stage of the plant to behave differently and realistically.

## **Recommended Workflow**

When starting with the plugin:

- Duplicate an existing Plant Prefab
- Modify:
  - Conditions
  - Actions
  - Threshold values
- Optionally replace visuals (sprites / meshes)

This approach allows you to create new plants without writing code, often only by adjusting data and visuals.

## **Full Extended Manual**

For a complete guide on creating your own plants, configuring PlantStates, actions, conditions, and prefabs in detail, please visit the Full Tutorial hosted on Notebook LM:

[Click here to access the Full Manual](#)

This extended guide provides step-by-step instructions, best practices, and example prefabs to help you fully leverage the plugin.

## **Community & Shared Content**

This plugin is designed to grow alongside its community.

The long-term goal is to provide:

- A shared Plant Prefab database
- A collection of reusable Soils, Plants, and configurations
- Community-driven improvements and discussions

All prefabs will be fully compatible with the simulation system and can be downloaded and used directly in your projects.

## **Discord Server**

<https://discord.gg/DPWHUP7vWA>

A dedicated Discord server is available for:

- Questions and troubleshooting
- Suggestions and feature requests
- Sharing Plant and Soil Prefabs
- Discussing simulation setups and design ideas

Community feedback will directly influence future updates of the plugin.